

IN01

Programmation Android

05 – Google Maps

Yann Caron
Et Jean-Marc Farinone

ENSG
Géomatique

Sommaire - Séance 05

- Géolocalisation – Principes
- Géolocalisation – Android
- Google Maps – Mise en œuvre
- Google Maps – Utilisation
 - ➔ Configurer, camera, markers, icon
 - ➔ Dessins, fenêtre
- Google API
 - ➔ Geocoding, Directions, Distance, Elevation
- Conclusion



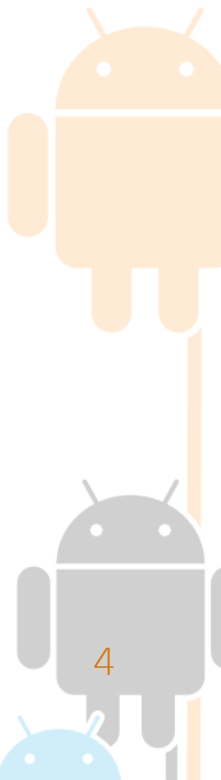
IN01 – Séance 05

Géolocalisation – principes



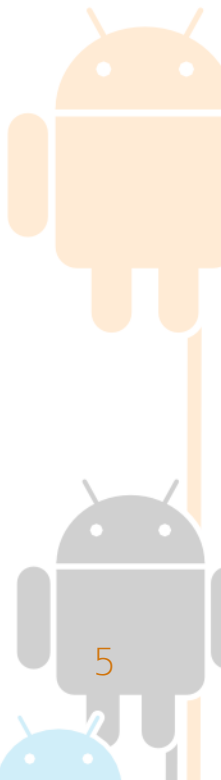
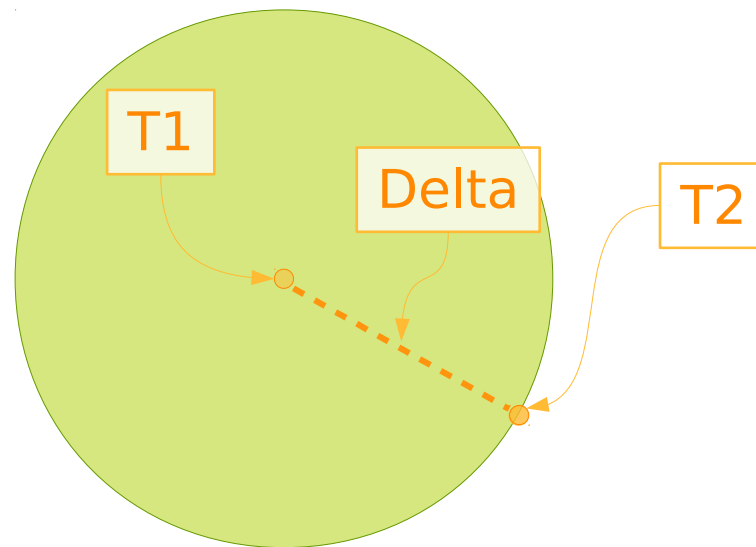
GPS

- GPS == Global Positioning System
- Un total de 24 satellites (21 + 3 secours) autour du globe répartis sur 6 orbites
- Utilise des satellites **NAVSTAR** (Navigation Satellite Timing And Ranging) diffusant leur position et l'heure
- Situés à **20 184 km** de la surface du globe, parcourent leur orbite en 12 h et émettent à des fréquences dans la bande des micro-ondes (**~1500Mhz**).



Temps de propagation

- Le GPS se synchronise puis calcule la distance avec le satellite grâce à la différence de temps en émission et réception
- Distance = $\Delta T * c$ (célérité du signal dans l'air ~ vitesse de la lumière : **$\sim 300\,000\text{ km/s}$**)



Triangulation

- Fonctionnement par triangulation : 4 positions sont nécessaires pour calculer la position.
- Un de plus par dimension souhaitée. 3D == 4 satellites

Avec 1 satellite :
circonférence

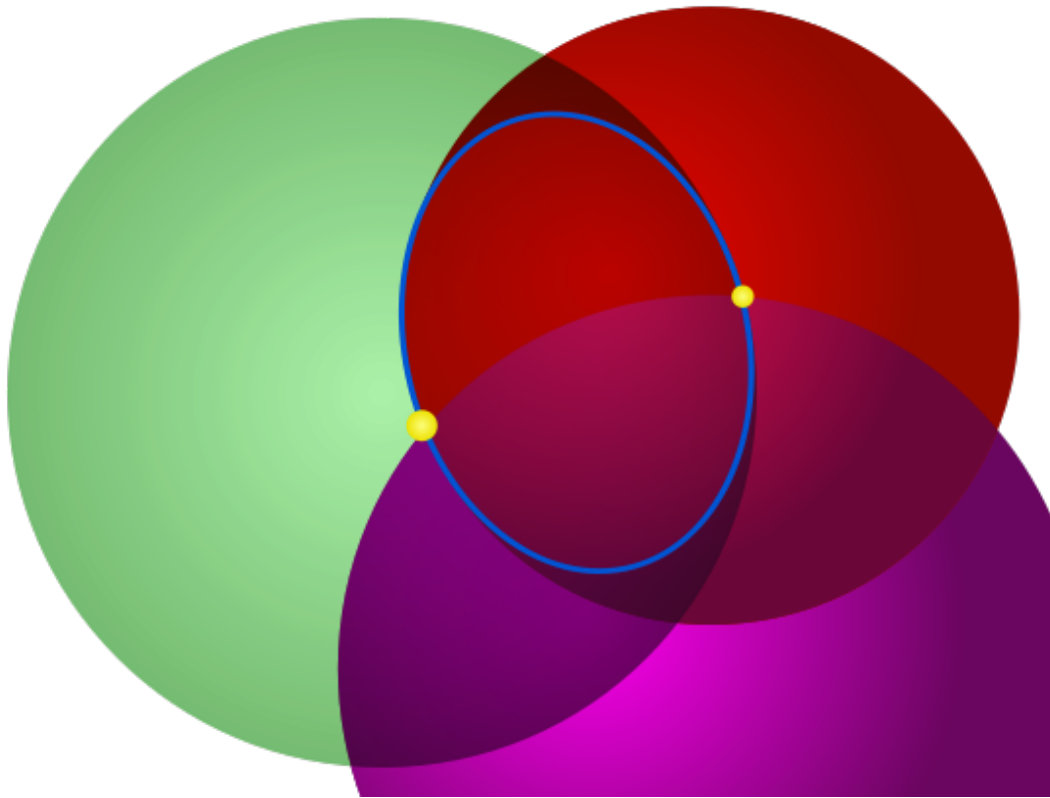
Avec 2 satellites :
2 points possibles

Avec 3 satellites : vous êtes ici !
Sur un plan à 2 dimensions

Mais à quelle altitude ??

Triangulation - Sphères

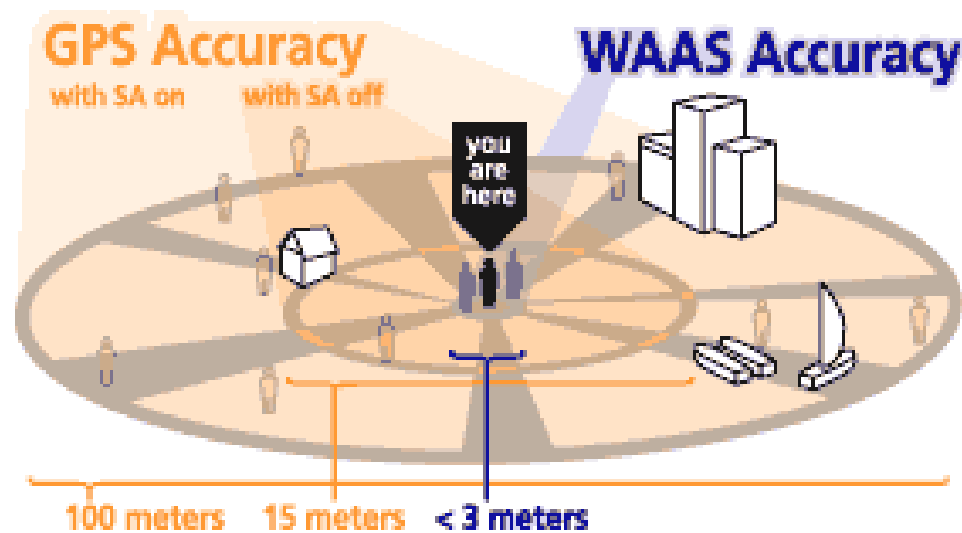
- En 3D, il faut imaginer des sphères :



Si les 2 points
sont sur la
surface de la
Terre, il faut un
4^e satellite

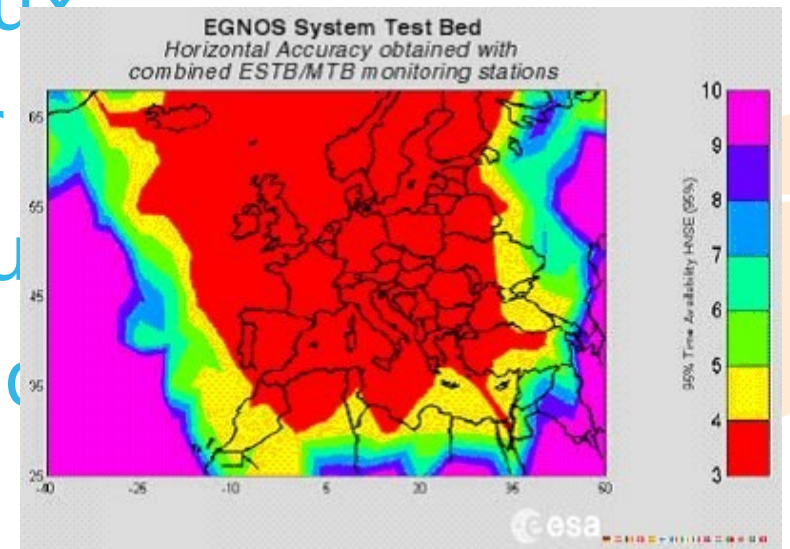
GPS - Précision

- Précision de 15 à 100 mètres
- WASS (US)/EGNOS (Eur) affinent la précision avec des balises au sol (< 3 mètres)



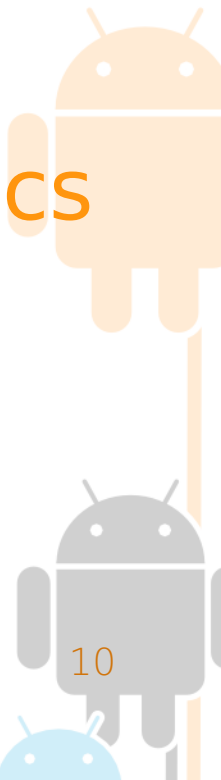
SBAS

- SBAS – Satellite-based Augmentation Systems
- Réseau de stations au sol (WMS Wide Area reference Station) qui connaissent leur position et la comparent aux coordonnées calculées par GPS. La correction est ensuite envoyée aux satellites via des satellites géostationnaires



Géolocalisation par relais

- La même chose, mais par relais téléphoniques GSM ou Wifi
- Conseillé, car consomme moins d'énergie (batterie) et la réception est meilleure
- Source :
<http://developer.android.com/guide/topics/location/strategies.html>

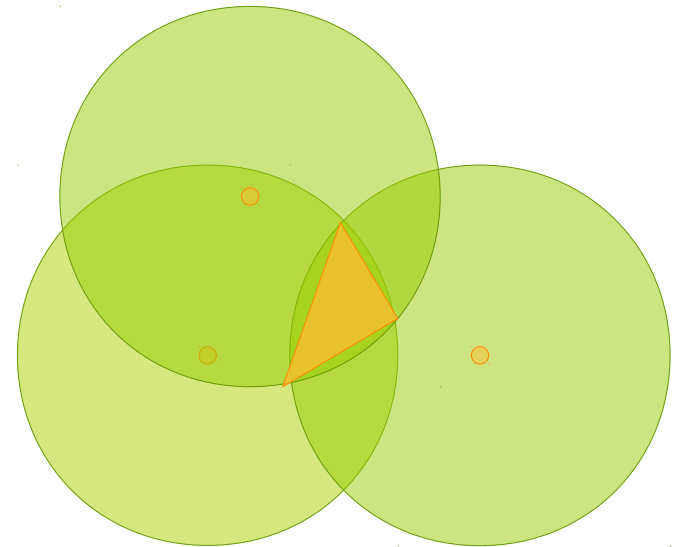


Géolocalisation CellID - GSM

- L'altitude est connue
- Fonctionne par **recouvrement** et non triangulation
- Cell-Id : évalue quels relais sont accessibles et déduit la zone de recouvrement



La combinaison des 3 antennes permettent d'obtenir les coordonnées XY du client



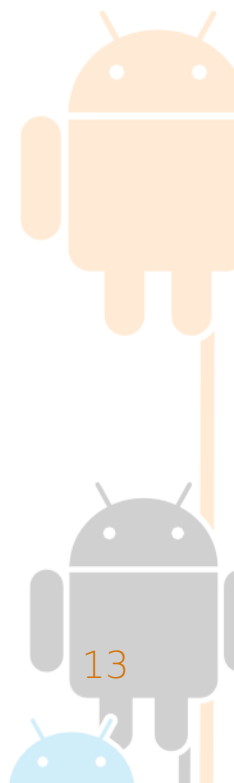
IN01 – Séance 05

Géolocalisation – Android



Géolocalisation - Android

- Pour utiliser la géolocalisation sur Android, il suffit de faire appel au bon service : `Context.LOCATION_SERVICE`
- Deux façons de procéder :
 - ➔ En récupérant la dernière position connue
 - ➔ De façon événementielle, c'est-à-dire recevoir une mise à jour régulière via un callback (système événementiel)



Dernière position connue

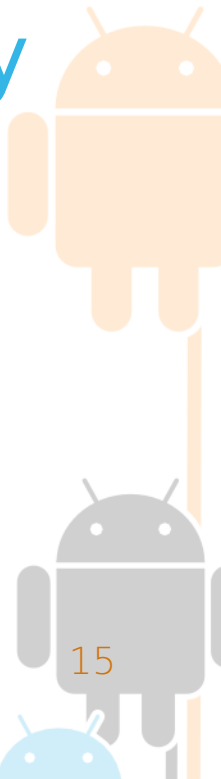
```
LocationManager locationManager =  
    (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);  
  
Criteria criteria = new Criteria();  
String provider = locationManager.getBestProvider(criteria, false);  
Location location = locationManager.getLastKnownLocation(provider);  
  
// Initialize the location fields  
if (location != null) {  
  
    float lat = (float) (location.getLatitude());  
    float lng = (float) (location.getLongitude());  
    Log.i(BaitFragment.class.getName(), String.valueOf(lat));  
    Log.i(BaitFragment.class.getName(), String.valueOf(lng));  
  
} else {  
  
    Log.i(BaitFragment.class.getName(), "Provider not available");  
    Log.i(BaitFragment.class.getName(), "Provider not available");  
  
}
```

Service

Dernière position connue

Géolocalisation Android - Callback

- On veut être notifié régulièrement pour traiter les changements
 - ➔ Par exemple pour tracer le déplacement sur une carte, calculer la vitesse, le déplacement
- Il suffit de le demander au service et d'y abonner un callback (évènement)



Géolocalisation Android - Callback

```
LocationManager locationManager =  
    (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
```

Service

```
LocationListener locationManager = new LocationListener() {  
    public void onLocationChanged(Location location) {}
```

Appelé lorsque la position change ou périodiquement

```
    public void onStatusChanged(String provider, int status, Bundle extras) {}
```

```
    public void onProviderEnabled(String provider) {}
```

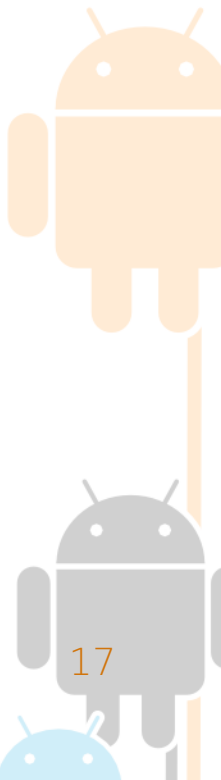
```
    public void onProviderDisabled(String provider) {}  
};
```

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10, 1,  
locationListener);
```

Attache le callback

Géolocalisation – Permissions

- Deux providers : `NETWORK_PROVIDER` (GSM et Wifi) et `GPS_PROVIDER`
- Il faut ajouter des autorisations à l'application
 - ➔ `ACCESS_COARSE_LOCATION` : pour la localisation à base de réseaux
 - ➔ `ACCESS_FINE_LOCATION` : comme la précédente, le GPS en plus



IN01 – Séance 05

Google Maps – Mise en œuvre



Google Maps

- L'avantage d'Android est de bénéficier d'applications développées par Google... dont Google Maps
- Une API est mise à disposition gratuitement depuis le SDK Manager.
- Il suffit de la télécharger et de demander une clé (API Key)
- Référence :
<https://developers.google.com/maps/documentation/android>



Google Maps

- On utilise donc les Google Maps Android API v2
- Cette API permet de manipuler des cartes terrestres
- Ces classes se trouvent dans le package `com.google.android.gms.maps`
- Pour afficher une carte, on utilise soit un fragment soit une vue
- Cette API gère les entrées clavier, le zoom, le toucher sur une carte affichée
- On peut dessiner, ajouter des images, des marqueurs et des infos sur la carte

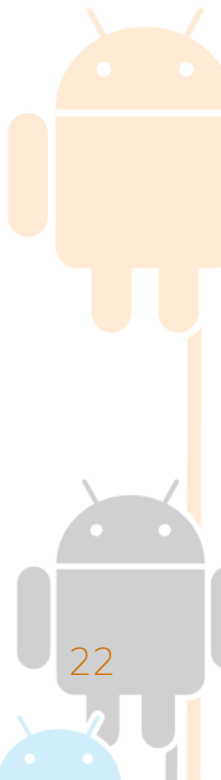
Google Maps

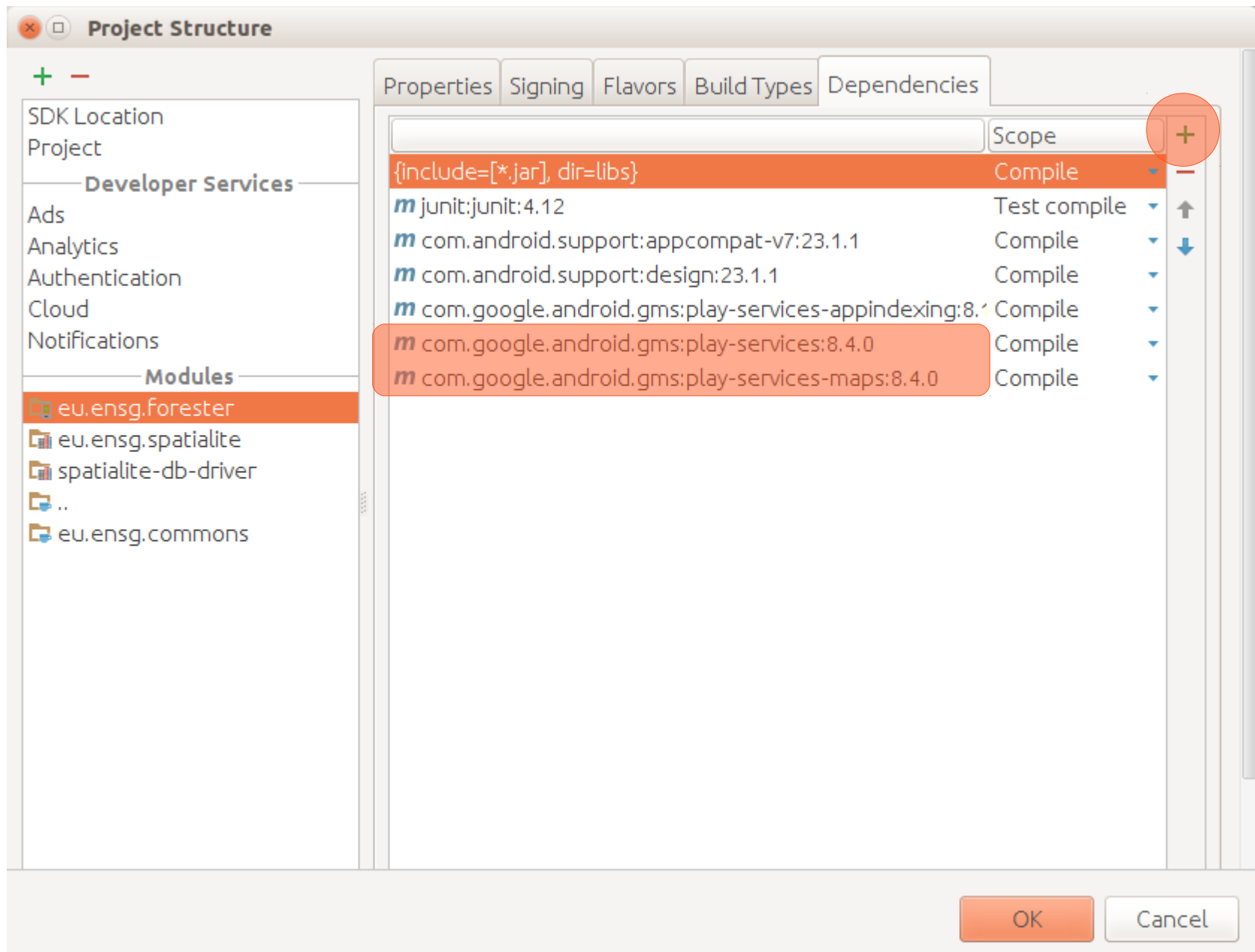
- La création d'une activity GoogleMap a été simplifié dans Android Studio
- Pour utiliser les cartes Google, il faut suivre les étapes suivantes :
 - 1 - Importer la bibliothèque googleMap dans le projet
 - 2 - Écrire une activité demandant à afficher une carte Google
 - 3 - Obtenir une clé Maps API v2 Key
 - 4 - Configurer le manifest
- Toute la procédure est indiquée à <https://developers.google.com/maps/documentation/android/start>



1 - Ajouter les dépendances

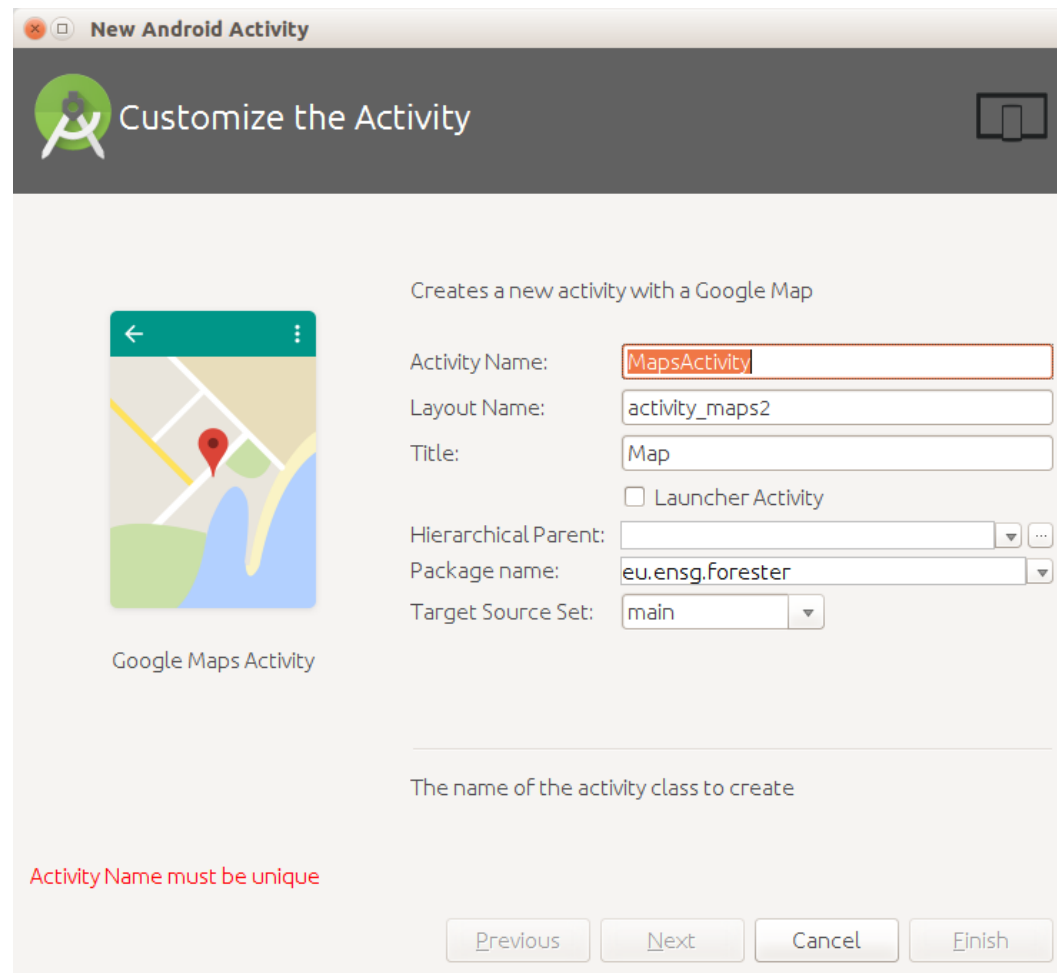
- Clique droit sur le projet
- OpenModuleSettings
- Onglet Dependencies
- Bouton “+”
- Library dependency
- Choisir `com.google.android.gms:`
 - ➔ `play-services:[version]`
 - ➔ `Play-services-maps:[version]`





2 – Créer une MapsActivity

- Dans Android Studio;
file/new/google/google map activity



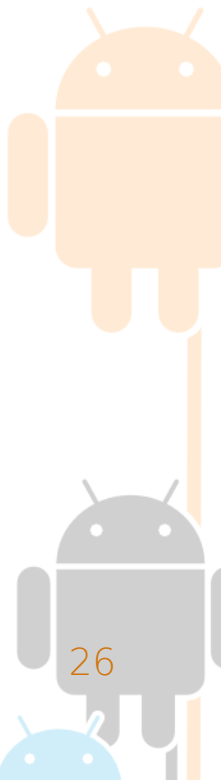
3 – Obtenir une clé

- Une fois l'activity créée il faut lancer l'application
- Dans les logs de l'application, GoogleMap indique une erreur :
 - ➔ Authentication failed on the server
- Suivi de l'url à contacter
 - ➔ <https://console.developers.google.com>
- Et du fingerprint
 - ➔ Un algorithme de hashage SHA-1 qui prend, entre autre, le numéro de série de l'appareil, des données de la machine de développement et le nom de l'application



3 – Obtenir une clé

- Se rendre sur le site developers.google.com (URL ci-dessus)
- Activer une nouvelle API
- Choisir GoogleMap Android API
- Cliquer sur Identifiants
- Créer une clé Android



Google Developers Console

- Y introduire le nom du package et le fingerprint

Créer une clé pour l'API Android

Nom

Limiter l'utilisation de vos applications Android (Facultatif)

Les requêtes API nous sont directement envoyées par les appareils Android. Nous vérifions que chaque requête provient d'une application Android dont le nom de package et l'empreinte de signature SHA-1 correspondent à ce que vous avez indiqué. Pour connaître le nom du package, reportez-vous au fichier AndroidManifest.xml, et pour obtenir l'empreinte, utilisez la commande suivante. [En savoir plus](#)

```
keytool -list -v -keystore mystore.keystore
```

Nom du package

Empreinte du certificat SHA-1

×

+ Ajouter nom du package et empreinte

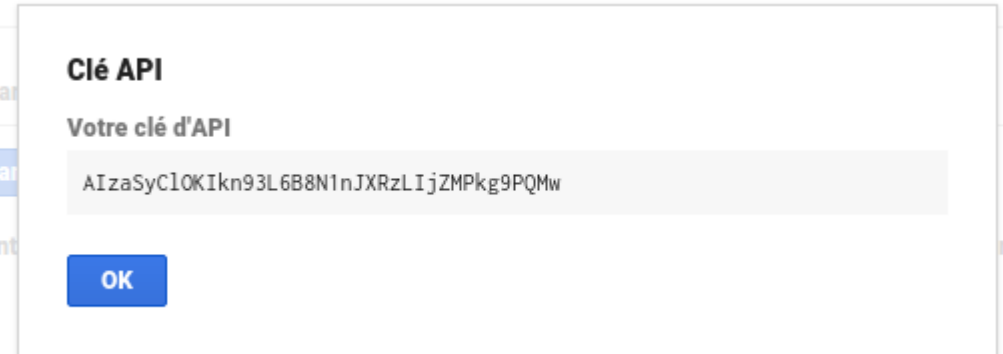
Remarque : L'application de ce paramètre peut prendre jusqu'à cinq minutes.

Créer

Annuler

3 – Optenir une clé

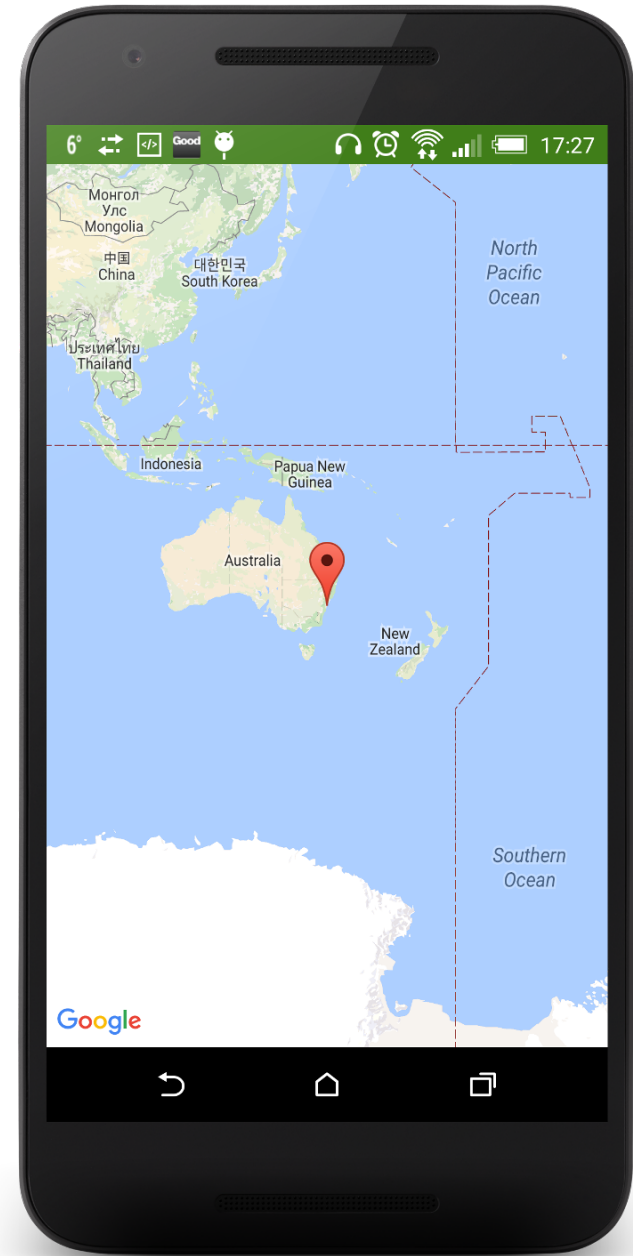
- Le site génère la clé
- Qu'il faut reporter dans le manifest de l'application :
`manifests/AndroidManifest.xml`



```
<meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="AIzaSyClOKIkn93L6B8N1nJXRzLIjZMPkg9PQMw" />
```

GoogleMap

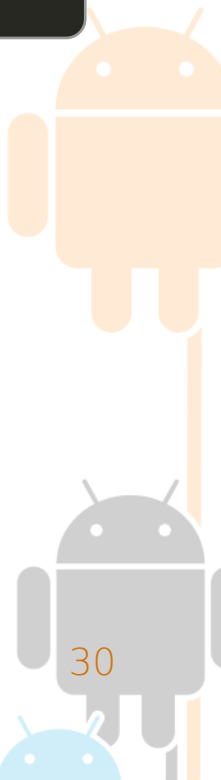
- Ou dans un fichier dédié
 - Res/values/google_maps_api.xml
- Lancer l'application



4 – Configurer AndroidManifest.xml

- Il faut ensuite créer quelques permissions standard

```
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />  
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```



4 – Configurer AndroidManifest.xml

- Ainsi que quelques paramètres optionnels :

```
<uses-permission  
  android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<uses-permission  
  android:name="android.permission.ACCESS_FINE_LOCATION" />
```

• Pour le rendu 3D

```
<uses-feature android:glEsVersion="0x00020000"  
  android:required="true"/>
```

4 – Manifest - Extrait

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

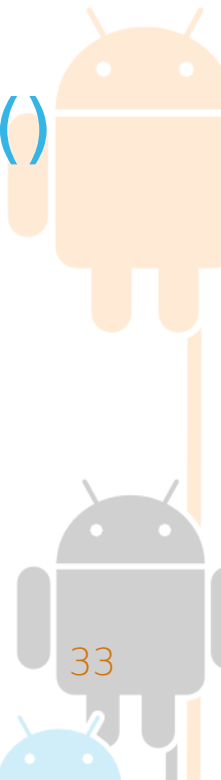
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />

<application
    android:allowBackup="true" android:icon="@drawable/ic_launcher"
    android:label="@string/global_app_name"
    android:theme="@style/AppTheme" >
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyA05eGksIpKNF7kUTuG5Qe3dCCIBG_hXKc" />
    </application>

</manifest>
```


Utiliser MapView

- Il est possible d'utiliser GoogleMap sous forme d'une vue : MapView
- Attention !! Si cette vue est utilisée, il faut relayer tous les évènements du conteneur (Activity, Fragment) vers la vue
- C.-à-d. : onCreate(), onDestroy(), onResume() et onPause()
- Sinon ça n'affiche rien (mauvaise expérience !!)



Dans le fichier XML

- On peut désormais ajouter la vue dans une Activity ou un Fragment. Ce sera un fragment.

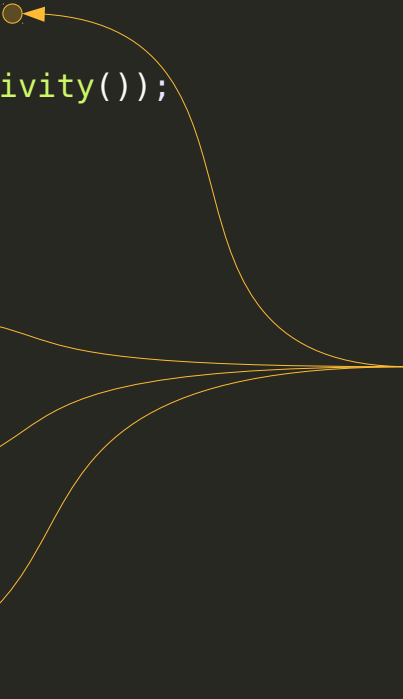
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <com.google.android.gms.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```

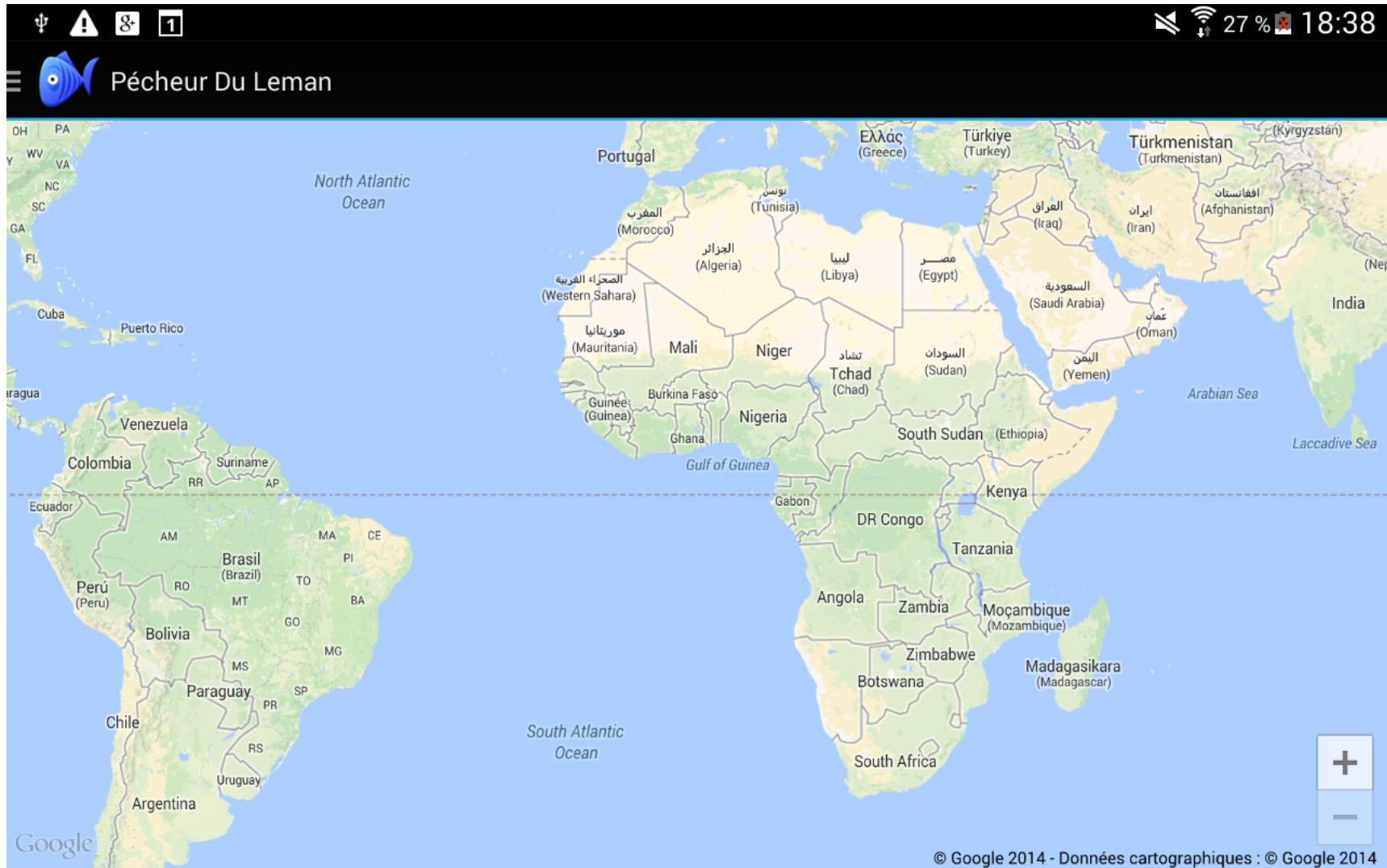
Dans le fichier Java

```
public class PlaceMapFragment extends Fragment {  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
        View v = inflater.inflate(R.layout.fragment_map, container, false);  
  
        mapView = (MapView) v.findViewById(R.id.mapview);  
        mapView.onCreate(savedInstanceState);  
  
        MapsInitializer.initialize(this.getActivity());  
  
        return v;  
    }  
  
    @Override  
    public void onResume() {  
        mapView.onResume();  
        super.onResume();  
    }  
  
    @Override  
    public void onDestroy() {  
        super.onDestroy();  
        mapView.onDestroy();  
    }  
  
    @Override  
    public void onLowMemory() {  
        super.onLowMemory();  
        mapView.onLowMemory();  
    }  
}
```

On relaye les évènements

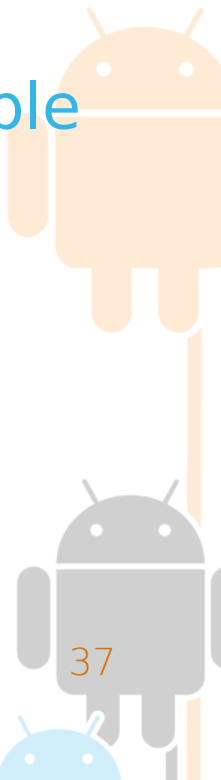


Résultat



Fragment

- Il est également possible d'utiliser googleMap sous la forme d'un Fragment
 - ➔ Android Responsive; gérer les tailles d'écrans
- Pour se faire, la classe GoogleMap doit hériter de SupportMapFragment ou MapFragment
 - ➔ Rend l'utilisation du composant graphique réutilisable
- Attention toutefois à la méthode onCreateView où il faut impérativement appeler la méthode surchargée



Fragment - code

```
public class MyMapFragment extends SupportMapFragment {  
  
    @Override  
    public View onCreateView(final LayoutInflater inflater, ViewGroup group,  
                             Bundle bundle) {  
        View view;  
        // TODO !!!! Invoquer le onCreateView de la super classe, sinon exception  
        // plutôt que : view = inflater.inflate(R.layout.map_fragment, group, false);  
  
        view = super.onCreateView(inflater, group, bundle);  
  
        this.getMapAsync(new OnMapReadyCallback() {  
            @Override  
            public void onMapReady(GoogleMap googleMap) {  
            }  
        });  
        return view;  
    }  
}
```

Attention

IN01 – Séance 05

Google Maps – Utilisation



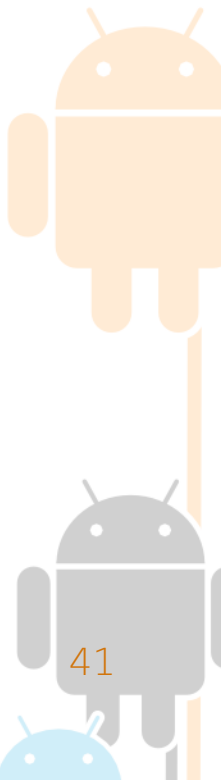
Types de Map

- Il existe plusieurs types de maps : NORMAL, HYBRID, SATELLITE, TERRAIN, NONE (juste la grille)
- Propriété Java :
`setMapType(GoogleMap.MAP_TYPE_NORMAL)`
- Tag XML : `mapType`
- Attention ! Il faut ajouter le xml name space (xmlns) dans la vue ou le layout XML :

```
xmlns:map="http://schemas.android.com/apk/res-auto"
```


Propriétés importantes

- **Position initiale de la caméra :**
`cameraTargetLat`, `cameraTargetLng`,
`cameraZoom`, `cameraBearing`, `cameraTilt`
- **Contrôles visuels :** `uiZoomControls`,
`uiCompass`
- **Comportement de la gesture :**
`uiZoomGestures`, `uiScrollGestures`,
`uiRotateGestures`, `uiTiltGestures`



Exemple

```
<com.google.android.gms.maps.MapView
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    map:cameraBearing="112.5"
    map:cameraTargetLat="-33.796923"
    map:cameraTargetLng="150.922433"
    map:cameraTilt="30"
    map:cameraZoom="13"
    map:mapType="hybrid"
    map:uiCompass="false"
    map:uiRotateGestures="true"
    map:uiScrollGestures="true"
    map:uiTiltGestures="true"
    map:uiZoomControls="true"
    map:uiZoomGestures="true" />
```

Namespace

Placer la caméra

- Un site web bien utile : <http://www.gps-coordinates.net/> pour obtenir les coordonnées géographiques

Address

DD (decimal degrees)*

Latitude

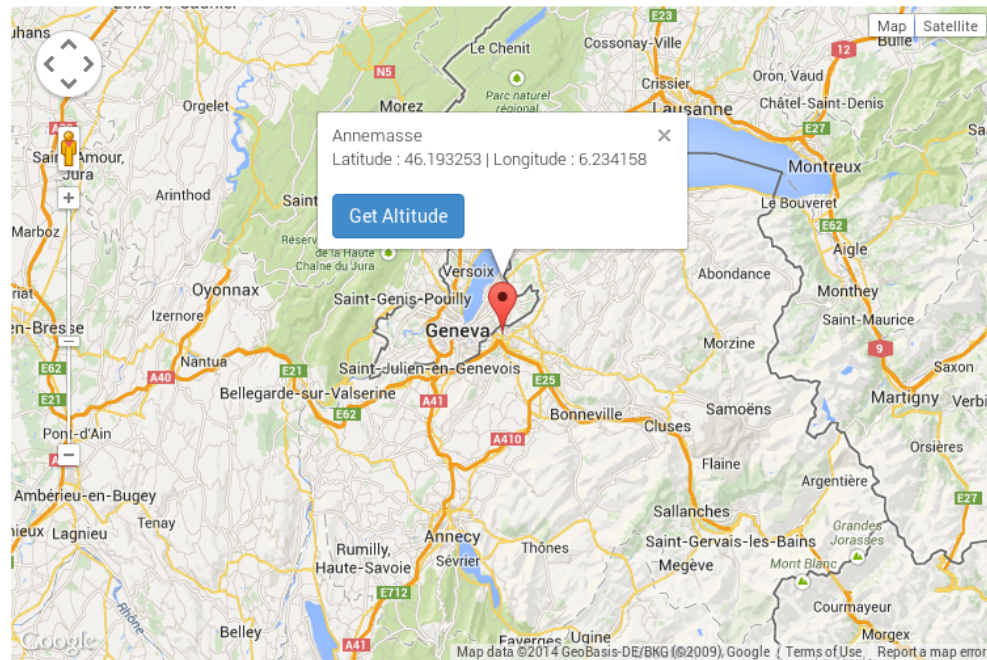
Longitude

DMS (degrees, minutes, secondes)*

Latitude ☐ N ☐ S ° ' "

Longitude ☐ E ☐ W ° ' "

* World Geodetic System 84 (WGS 84)



Placer la caméra

- Le placement est animé
- Il faut créer un objet LatLng (un tuple latitude et longitude)
- Ou utiliser la dernière position connue (GPS ?)

```
MapsInitializer.initialize(this.getActivity());  
  
// position  
LatLng annemasse = new LatLng(46.193253, 6.2341579);  
  
// positionnement initial  
map.moveCamera(CameraUpdateFactory.newLatLngZoom(annemasse, 15));  
  
// animation  
map.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
```

Obligatoirement avant

Zoom initial

Type d'animation et données d'arrivée

Durée de l'animation

Ajouter un marqueur

- Marqueur par défaut
- On définit une position et un titre
- La propriété `snippet()`, c'est le texte qui apparaît lorsque l'utilisateur clique sur le marqueur

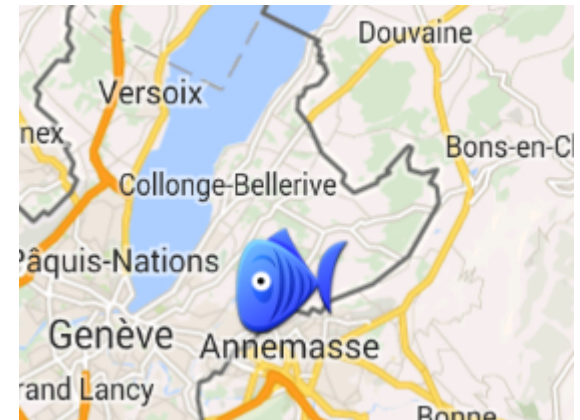
```
map.addMarker(new MarkerOptions()  
    .position(annemasse)  
    .title("Ma maison")  
    .snippet("J'habite Annemasse")  
);
```



Marqueur personnalisé

- Plusieurs propriétés permettent de personnaliser le marqueur

➔ `icon()`, `alpha()`,
`rotation()`, `draggable()`



```
map.addMarker(new MarkerOptions()  
    .position(annemasse)  
    .title("Ma maison")  
    .snippet("J'habite Annemasse")  
    .icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_launcher))  
);
```

Custom info windows

- On peut personnaliser la fenêtre d'informations (titre et snippet)



Custom info windows

- Comme pour toute UI, il est possible de tout écrire en Java ou en XML (inflater)
- Il faut prévoir des TextView pour le titre et le texte du snippet

```
<android.support.v7.widget.GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res/fr.cnam.pecheurduleman"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:columnCount="2" >

    <ImageView
        android:src="@drawable/fish_green"
        app:layout_rowSpan="2" />

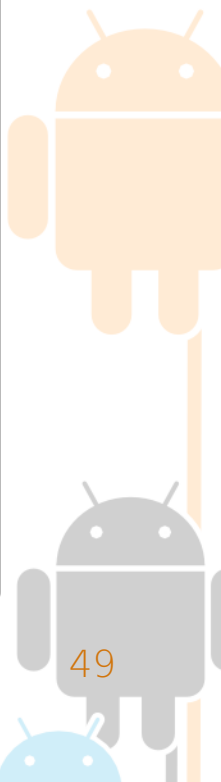
    <TextView
        android:id="@+id/info_title"
        android:layout_margin="5dp"
        android:textSize="20dp"
        android:textColor="#ff00760f" />

    <TextView
        android:id="@+id/info_snippet"
        android:layout_margin="5dp"
        android:textSize="15dp"
        android:textColor="#ff303030" />
</android.support.v7.widget.GridLayout>
```


Custom info windows

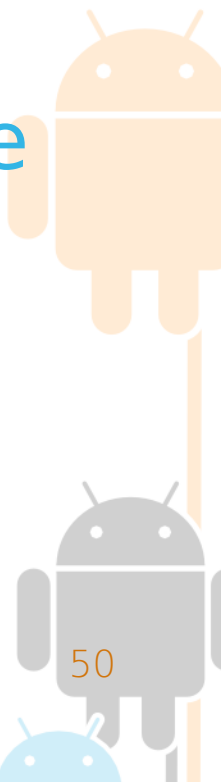
- Il suffit d'implémenter l'interface `InfoWindowAdapter`

```
map.setInfoWindowAdapter(new InfoWindowAdapter() {  
  
    @Override  
    public View getInfoWindow(Marker marker) {  
        return null;  
    }  
  
    @Override  
    public View getInfoContents(Marker marker) {  
        View v = inflater.inflate(R.layout.map_info_window, null);  
  
        TextView title = (TextView) v.findViewById(R.id.info_title);  
        title.setText(marker.getTitle());  
  
        TextView snippet = (TextView) v.findViewById(R.id.info_snippet);  
        snippet.setText(marker.getSnippet());  
  
        return v;  
    }  
});
```



InfoWindowAdapter


- Deux méthodes à implémenter :
 - ➔ `getInfoWindow()` : permet de customiser toute la fenêtre
 - ➔ `getInfoContents()` : permet de ne customiser que le contenu de la fenêtre
- Attention !! Dans le cas où on implémente `getInfoContents()`, la vue est chargée comme une image statique (pas de changement possible)



Plus d'informations

- `setOnInfoWindowClickListener()` : comme son nom l'indique, permet de brancher un callback sur le clic de l'info window

```
map.setOnInfoWindowClickListener(new OnInfoWindowClickListener() {  
  
    @Override  
    public void onInfoWindowClick(Marker marker) {  
        Log.i(PlaceMapFragment.class.getName(),  
            "On a cliqué sur " +  
            marker.getTitle() +  
            " [" + marker.getSnippet() + "]);  
    }  
});
```



Dessiner sur la carte

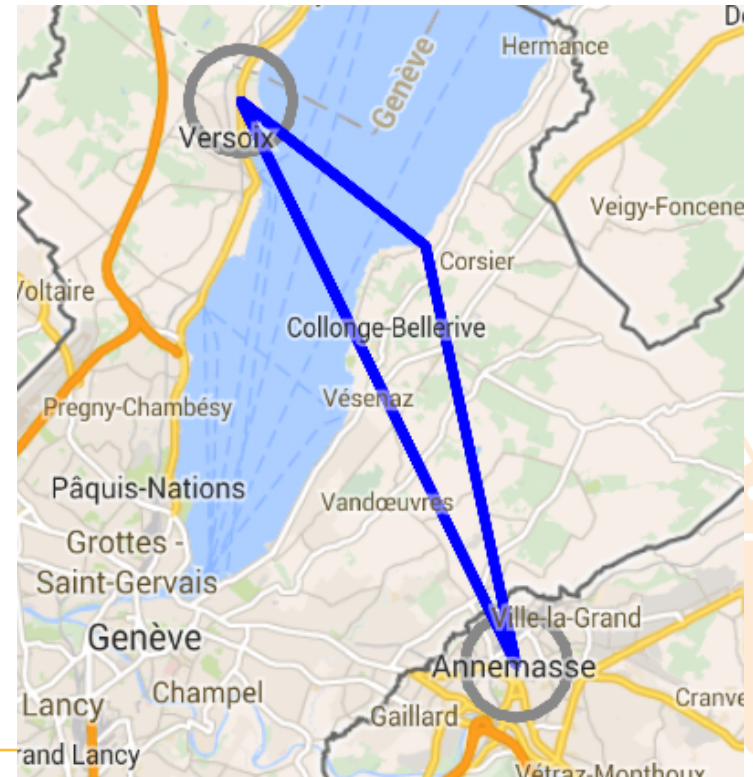
- Il est possible de dessiner des formes géométriques sur la carte selon des coordonnées géographiques
 - ➔ Des cercles : `addCircle()`
 - ➔ Des polygones : `addPolygone()`
 - ➔ Des polylines (polygone vide) : `addPolyline()`
 - ➔ Des images : `addGroundOverlay()`
 - ➔ Des textures : `addTileOverlay()`. Ce dernier permet de décorer ou de remplacer les images de terrain fournies par Google. Il faut gérer les niveaux de zoom



Exemple de dessin

```
map.addCircle(new CircleOptions()  
    .center(annemasse)  
    .radius(1000) // en mètres  
    .strokeWidth(5).strokeColor(Color.GRAY));  
  
map.addCircle(new CircleOptions()  
    .center(portChoiseul)  
    .radius(1000) // en mètres  
    .strokeWidth(5).strokeColor(Color.GRAY));  
  
map.addPolyline((new PolylineOptions()  
    .add(annemasse, portChoiseul,  
        corsierPort, annemasse)  
    .width(5).color(Color.BLUE)  
    .geodesic(true));
```

Coordonnées LatLng



IN01 – Séance 05

Google API

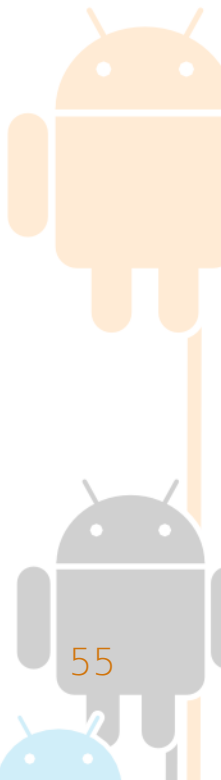


Google API



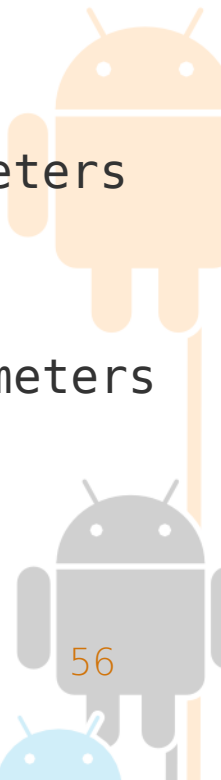
Services Web

- Depuis quelques années, Google met à disposition des Webservices dédiés à GoogleMap
- <https://developers.google.com/maps/web-services/>
- Notamment : Geocoding, Places, Elevations, Geolocation, Directions



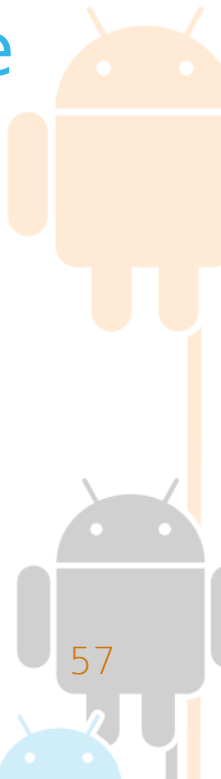
Principe

- L'application android appelle une URL
- Le serveur HTTP lui renvoie un “streaming” XML ou JSON
- Détaillé dans le cours sur les WebServices
- URL HTTP :
 - ➔ `http://maps.googleapis.com/maps/api/geocode/output?parameters`
- URL HTTPS :
 - ➔ `https://maps.googleapis.com/maps/api/geocode/output?parameters`



Google Maps Geocoding API

- Destiné à affecter des coordonnées géographiques à des adresses.
 - ➔ Exemple : "1600 Amphitheatre Parkway, Mountain View, CA"
 - ➔ Renvoie : latitude 37.423021 and longitude -122.083739
- Et inversement "reverse geocoding"



Geocoding

Address

Eden Gardens, Kolkata

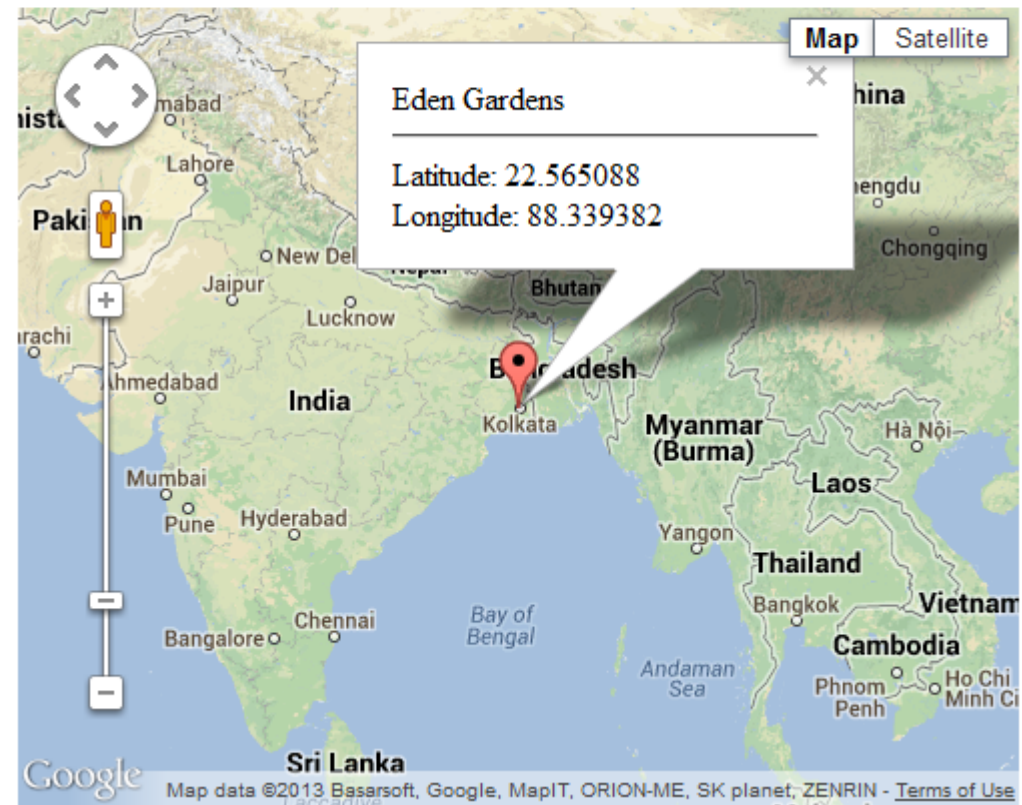
Find

Latitude

22.565088

Longitude

88.339382



Exemple

- JSON URL :

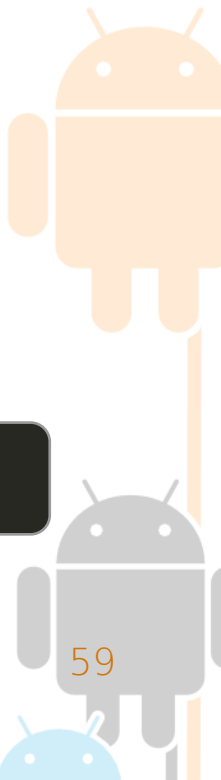
- ➔ `https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY`

- XML URL :

- ➔ `https://maps.googleapis.com/maps/api/geocode/xml?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY`

- YOUR_API_KEY (bonne pratique) :

```
getResources().getString(R.string.google_maps_key);
```



JSON

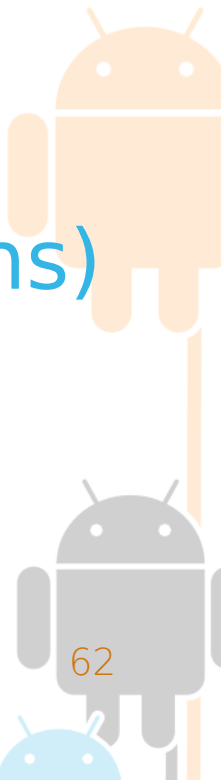
```
{
  "results" : [
    {
      "address_components" : [
        { "long_name": "1600", "short_name": "1600", "types": ["street_number"] },
        [...]
      ],
      "formatted_address" : "1600 Amphitheatre Parkway, Mountain View, CA
94043, USA",
      "geometry" : {
        "location" : { "lat" : 37.4224764, "lng" : -122.0842499 },
        "location_type" : "ROOFTOP",
        "viewport" : {
          "northeast" : { "lat" : 37.4238253802915, "lng" : -122.0829009197085 },
          "southwest" : { "lat" : 37.4211274197085, "lng" : -122.0855988802915 }
        }
      },
      "place_id" : "ChIJ2eUgeAK6j4ARbn5u_wAGqWA",
      "types" : [ "street_address" ]
    }
  ],
  "status" : "OK"
}
```

XML

```
<GeocodeResponse>
  <status>OK</status>
  <result>
    <type>street_address</type>
    <formatted_address>1600 Amphitheatre Pkwy, Mountain View, CA 94043,
USA</formatted_address>
    <geometry>
      <location>
        <lat>37.4217550</lat>
        <lng>-122.0846330</lng>
      </location>
      <location_type>R00FT0P</location_type>
      <viewport>
        <southwest>
          <lat>37.4188514</lat>
          <lng>-122.0874526</lng>
        </southwest>
        <northeast>
          <lat>37.4251466</lat>
          <lng>-122.0811574</lng>
        </northeast>
      </viewport>
    </geometry>
    <place_id>ChIJ2eUgeAK6j4ARbn5u_wAGqWA</place_id>
  </result>
</GeocodeResponse>
```

Reverse Geocoding

- Address Lookup
- Determine l'adresse à partir de la position
 - ➔ `https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key=YOUR_API_KEY`
- Le résultat est le même que pour le Geocoding (contient les deux informations)



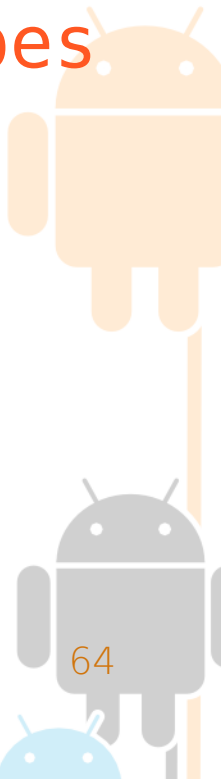
Google Maps Geolocation

- Google Maps Geolocation donne la position de l'appareil en se basant sur les antennes relais et les bornes Wi-fi
- Utile si l'appareil ne connaît pas sa position
- Google sait où vous êtes
- URL :
 - ➔ `https://www.googleapis.com/geolocation/v1/geolocate?key=YOUR_API_KEY`



Google Places API Web Service

- Permet de rechercher les lieux intéressants
- Recherche aux alentours :
 - ➔ `https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=-33.8670522,151.1957362&radius=500&types=food&name=cruise&key=YOUR_API_KEY`
- Renvoie une liste de placeID
 - ➔ Exemple : ChIJN1t_tDeuEmsRUsoyG83frY4



Google Places API Web Service

- **Détail du lieu :**

- ➔ `https://maps.googleapis.com/maps/api/place/details/
json?
placeid=ChIJN1t_tDeuEmsRUsoyG83frY4&key=YOUR_API_KEY`

- **Il faut utiliser le placeId**

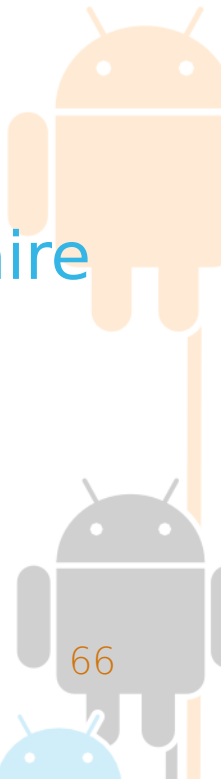
- **Mais aussi :**

- ➔ Ajout d'un lieu d'intérêt (modéré par Google)
 - ➔ Accès aux photos du lieu
 - ➔ Auto complétion (liste de lieux, selon les premières lettres saisies)



Google Maps API

- Elevation API : renvoie les altitudes entre deux points ou selon un chemin
- Google Maps Roads API : renvoie les routes traversées lors d'un trajet
- Google Maps Distance Matrix API : les distances et durées pour plusieurs destinations
- Google Maps Directions API : Calcul l'itinéraire
- Google Maps Time Zone API : renvoie le fuseau horaire de la position
- En plus de Geocoding, Geolocation et Places



Java Client for Google Maps Services

- Une API Java (OpenSource) pour abstraire la manipulation des web service Google
- <https://github.com/googlemaps/google-maps-services-java>
- Exemple :

```
GeoApiContext context = new GeoApiContext().setApiKey("AIza...");  
  
GeocodingResult[] results = GeocodingApi.geocode(context,  
    "1600 Amphitheatre Parkway Mountain View, CA 94043").await();  
  
System.out.println(results[0].formattedAddress);
```

IN01 – Séance 05

Conclusion

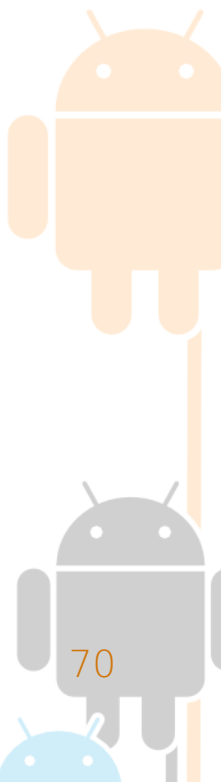


Conclusion

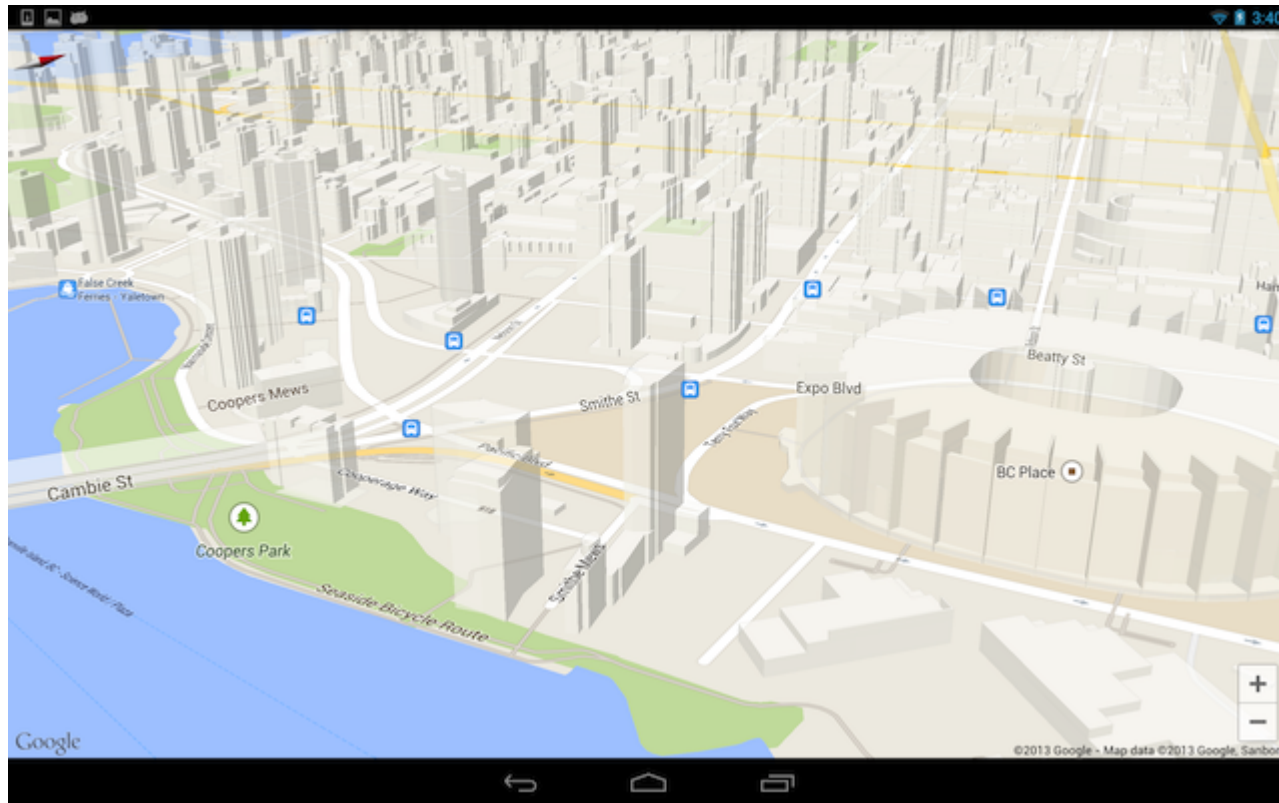
- Une vue très puissante. Un grand nombre de possibilités
- Utilisation de la géolocalisation (tracés temps réels, enregistrement de position)
- Un grand nombre de webservices fournis par Google
- Mais d'autres sont disponibles, quelques idées :
 - ➔ Windspot (afficher les vents), Météo, METAR (afficher les données météo aéronautiques par aéroports)
- D'autres idées ??

Pour aller plus loin

- Google met à disposition StreetView dans son API : `StreetViewPanoramaFragment`
- **My location button** `setMyLocationButtonEnabled` et `setOnMyLocationButtonClickListener`
- **Afficher les batiments** : `setBuildingEnabled`
- **Incliner la caméra** : `CameraPosition.Builder.bearing, target, tilt` et `zoom`
- **Évènements** : `setOnMapClickListener`



Batiments 3D



Fin

- Merci de votre attention
- Des questions ?

